SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Dominik Levanić

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:
Prof. dr. sc. Bojan Jerbić

Student:
Dominik Levanić

Zagreb, 2019.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem mentoru prof. dr. sc. Bojanu Jerbiću te asistentu dr. sc. Filipu Šuligoju na pomoći i sugestijama prilikom izrade završnog rada.

Naposljetku zahvaljujem svojim roditeljima, ostatku obitelji i prijateljima na podršci tijekom studija.

Dominik Levanić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: DOMINIK LEVANIĆ Mat. br.: 0035203743

Naslov rada na hrvatskom jeziku: **Vizijski sustav za praćenje objekata aktuiran servo motorom**

Naslov rada na engleskom jeziku: **Vision system for tracking objects actuated by servo motor**

Opis zadatka:

Primjena robota u sve kompleksnijim zadaćama podrazumijeva primjenu vizijskih sustava za lokalizaciju i praćenje objekata. Vizualno servo-navođenje je tehnika koja koristi povratne informacije dobivene iz vizijskog sustava za upravljanje mehaničkim sustavom ili robotom. Za potrebe završnog rada potrebno je:

1. Projektirati i integrirati sustav koji se sastoji od kamere, upravljačkog sklopa i servo motora.
2. Primijeniti algoritme strojnog vida za lokalizaciju pomičnih objekata u definiranom radnom prostoru.
3. Upravljati servo-motorom na temelju informacija dobivenih iz vizijskog sustava (centriranje pomičnog objekta).
4. Optimirati sustav s ciljem dobivanja veće brzine procesuiranja i odaziva.
5. Evaluirati izvedbu i mogućnosti primjene razvijenog sustava.

Zadatak zadan:
29. studenog 2018.

Rok predaje rada:
1. rok: 22. veljače 2019.
2. rok (izvanredni): 28. lipnja 2019.
3. rok: 20. rujna 2019.

Predviđeni datumi obrane:
1. rok: 25.2. - 1.3. 2019.
2. rok (izvanredni): 2.7. 2019.
3. rok: 23.9. - 27.9. 2019.

Zadatak zadao:

Prof. dr. sc. Bojan Jerbić

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Zadatak ovog završnog rada je projektirati i integrirati računalni vizijski sustav za lokalizaciju i praćenje objekata (eng. Object Localization and Tracking) prema nekoliko algoritama, a koji se sastoji od kamere, upravljačke jedinice Raspberry Pi te servo motora. Kamerom snimljen video zapis se obrađuje i u realnom vremenu prikazuje na zaslonu računala. Računalo je spojeno s upravljačkom jedinicom korištenom za upravljanje servo motorom koji pokreće kameru i u realnom vremenu je usmjerava prema određenom objektu. Na početku rada će se razjasniti osnovni pojmovi vezani uz područje vizijskih sustava te prednosti i nedostaci njihove uporabe. Zatim će biti opisane korištene komponente sustava, kao i razvijeni algoritmi korišteni za lokalizaciju i praćenje objekata. U razvoju algoritama korištena je računalna knjižnica otvorenog koda OpenCV. Računalni program je razvijen u programskom jeziku Python. Naposljetku će se evaluirati izvedba navedenog vizijskog sustava te dati prijedlog njegove moguće primjene.

Ključne riječi: vizijski sustav, Raspberry Pi, servo motor, OpenCV knjižnica

SUMMARY

The goal of this final project is to design and integrate a computer vision system for object localization and tracking according to a couple of algorithms, and which consists of a camera, a control unit Raspberry Pi and a servomotor. Video record captured by the camera is processed and in the real-time reproduced on the computer screen. Computer si connected with the control unit used for controlling servomotor which starts the camera and directs it in the real-time to the specified object. At the beginning of the work, the basic terms about computer vision will be clarified, as will their pros and cons as well. Then the used system components will be described, as will developed algorithms used for object localization and tracking. In the development of algorithms, the open-source library OpenCV is used. At last, the evaluation and the proposal of possible application of the cited vision system will be given.

Key words: vision system, Raspberry Pi, servomotor, OpenCV library

1. UVOD

Područje računalnih vizijskih sustava ili računalnog vida vrlo je slično područjima obrade i analize slike te području strojnog vida. Računalni vid je proces primjene niz metoda radi korištenja dohvaćenih informacija iz slika u svrhu prepoznavanja objekata, analize pokreta ili rekonstrukcije modela iz slika. Obrada i analiza slika je, s druge strane, proces primjene niza metoda radi otklanjanja šumova iz slika, dohvaćanja informacija iz slika ili priprema slika za daljnju obradu. Strojni vid je pak proces primjene niza metoda radi upravljanja slikovno baziranim procesima u industriji i njihove kontrole, a oslanja se na korištenje različitih senzora te industrijskih kamera za dohvaćanje slika. Slike mogu biti dohvaćene iz rendgenskog, vidljivog ili infracrvenog spektra svjetlosti. [1]

Računalni vid se fokusira na stvaranje autonomnog sustava koji bi mogao obavljati zadatke kakve može obavljati i ljudski vid ili ljudski vizijski sustav. Mnogo je takvih zadataka povezano s korištenjem informacija iz snimki ili s rekonstrukcijom modela iz slika. [2]

Strojni vid pokazuje prednosti nad ljudskim vidom kod zadataka povezanih s kvantificiranim mjerenjima u strukturiranoj sceni, a razlog tome su njegova brzina, preciznost i ponovljivost. Kod strojnog se vida korištenjem pravilno odabranog optičkog sustava mogu jednostavno izvoditi i zadaci nad vrlo malim predmetima, koji su premali za korištenje ljudskog vida. [3]

Svrha računalnog i strojnog vida je smanjenje proizvodnog vremena predmeta rada, smanjenje troškova, povećanje fleksibilnosti sustava, oslobađanje radnika monotonog rada te mogućnost korištenja istih na drugim radnim mjestima. [4]

Područje računalnih vizijskih sustava počelo se razvijati šezdesetih godina prošlog stoljeća na američkom sveučilištu „Massachusetts Institute of Technology“, dohvaćanjem informacija iz dvodimenzionalnih perspektiva predmeta i njihovim pretvaranjem u trodimenzionalne perspektive. Nakon nekog vremena uviđena je potreba za korištenjem navedenog pristupa u slikama iz stvarnog svijeta. [2]

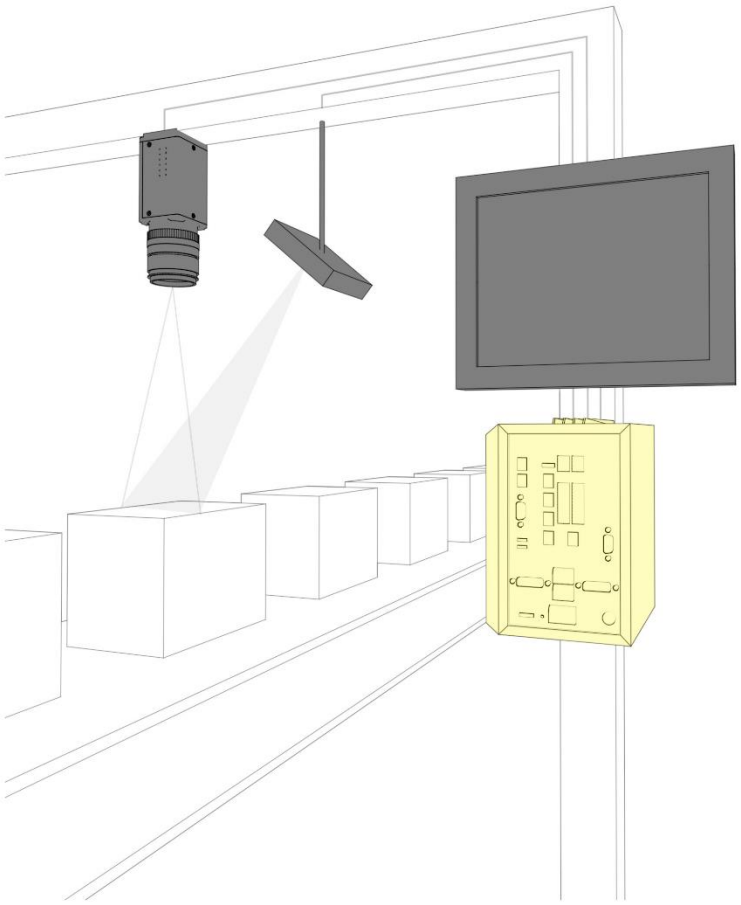
Mnoga su se istraživanja počela orijentirati na zadatke niske razine zahtjevosti, poput detekcije rubova predmeta sa slike i segmentacije slike te je sedamdesetih godina prošlog stoljeća na navedenom sveučilištu započet tečaj naziva „Machine Vision“ (hrv. strojni vid ili strojni vizijski sustav). Devedesetih godina prošlog stoljeća, strojni su se vizijski sustavi već počeli koristiti u industriji te su njihove cijene znatno padale. [5]

1.1. Dijelovi vizijskih sustava

Vizijski sustavi sastoje se od sljedećih komponenata: [3]

- optičkog senzora
- leće kamere
- računala za obradu slike
- rasvjete.

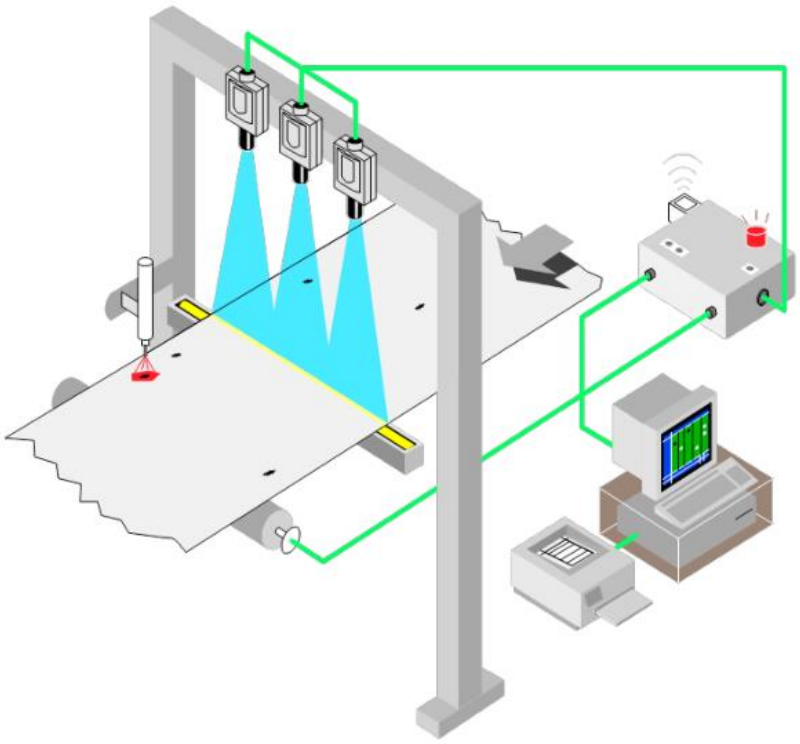
Rasvjeta omogućava isticanje onih značajki predmeta rada koje trebaju biti zapažene u slici dohvaćenoj kamerom. Slika se dohvaća kamerom tako da leća kamere dohvaća informacije o svjetlosti koje se zatim u optičkom senzoru pretvaraju u digitalne signale koji zajedno prikazuju sliku. Slike se obrađuju nizom metoda pomoću računala te se njihovi rezultati prikazuju na izlaznoj jedinici. [3]



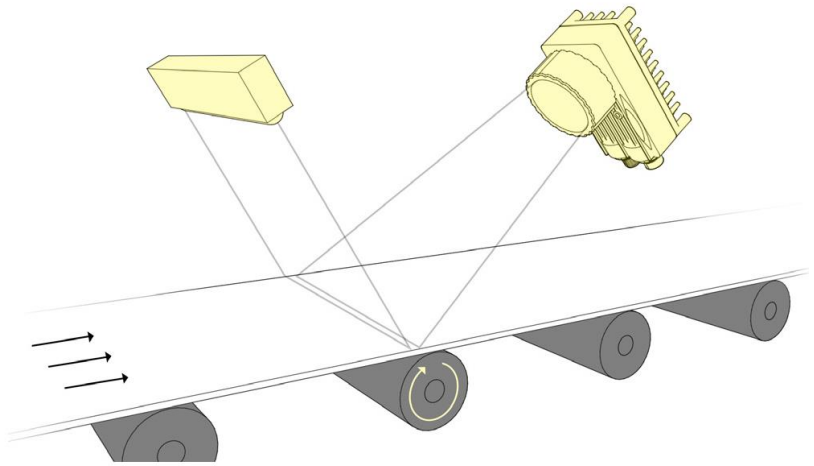
Slika 1. Shematski prikaz vizijskog sustava

1.2. Vrste vizijskih sustava

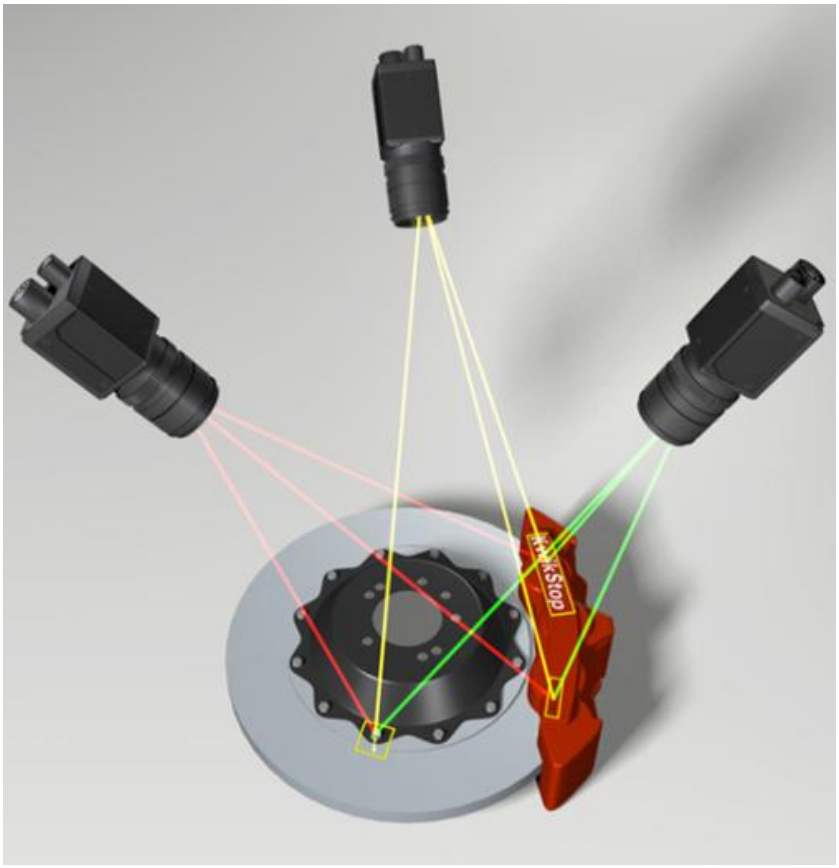
Vizijski se sustavi mogu podijeliti na jednodimenzionalne, dvodimenzionalne i trodimenzionalne. Jednodimenzionalni vizijski sustavi analiziraju digitalni signal jedne linije te su izvedeni s laserskim senzorom. Ta se tehnologija najčešće koristi kod detekcije i klasifikacije pogrešaka na predmetima rada kontinuiranog procesa. Dvodimenzionalni vizijski sustavi analiziraju digitalne signale u dvije dimenzije, a mogu biti izvedeni s kamerom tako da dohvaćaju informacije o obje dimenzije odjednom ili s laserskim senzorom tako da dohvaćaju informacije o jednoj po jednoj liniji. Trodimenzionalni vizijski sustavi analiziraju digitalne signale u tri dimenzije, a mogu biti izvedeni s dvije ili više kamere tako da dohvaćaju informacije o sve tri dimenzije odjednom ili s laserskim senzorom pomičnim u odnosu na predmet rada tako da dohvaćaju informacije o jednoj po jednoj liniji i o njenoj visini. [3]



Slika 2. Shematski prikaz jednodimenzionalnog vizijskog sustava izvedenog s laserskim senzorom



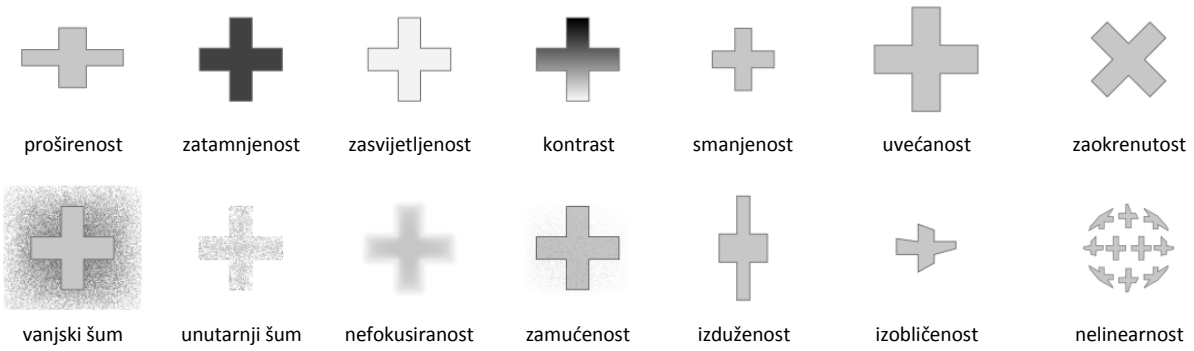
Slika 3. Shematski prikaz dvodimenzionalnog vizijskog sustava izvedenog s laserskim senzorom



Slika 4. Shematski prikaz trodimenzionalnog vizijskog sustava izvedenog s tri kamere

1.3. Prednosti i nedostaci

Područja strojnih vizijskih sustava sama su po sebi vrlo zahtjevna područja, pošto se za rijetko koji zadatak može pronaći potpuno zadovoljavaće rješenje. Glavni je uzrok tomu to što je ljudski vid mnogo bolji od strojnog za većinu zadataka, osim kod zadataka povezanih s kvantificiranim mjerenjima u strukturiranoj sceni. Ljudski vid može prepoznati objekte pod različitim osvjetljenjem, iz različite perspektive, kod različite zaklonjenosti i bez poznatog ograničenja u broja objekata za pamćenje. U prilog navedenome, ljudski vid može prepoznati djelomično zaklonjenu osobu na slici uslikanoj prije mnogo godina. [2]

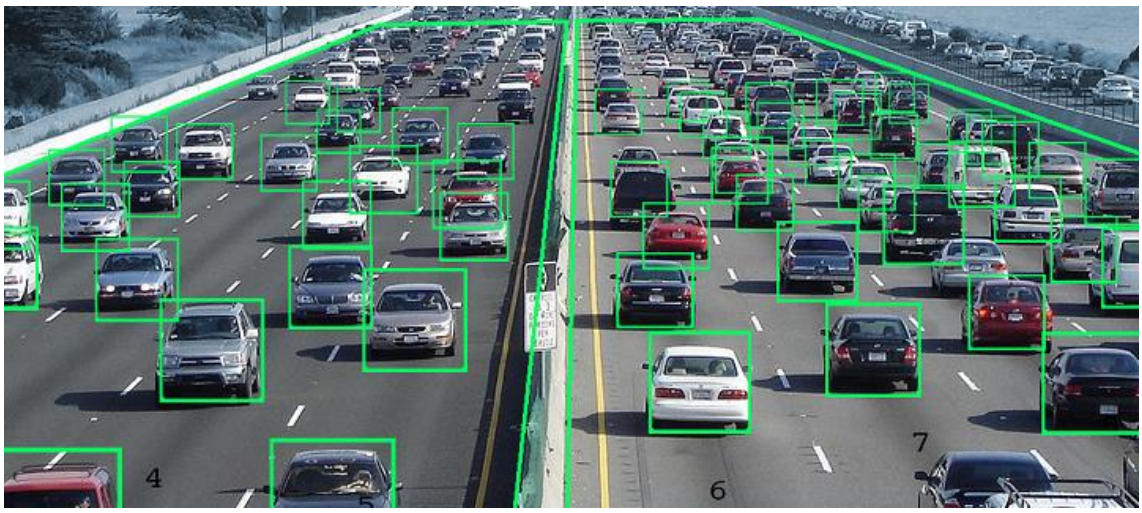


Slika 5. Mogući razlozi neprepoznavanja objekata strojnim vidom

Postavlja se pitanje kako pomoću računala filtrirati i strukturirati ogromnu količinu informacija prikupljenih iz sveukupnog ljudskog znanja na način da se tražene informacije mogu lako dohvatiti. Na navedeno se nadovezuje i pitanje načina hardverske i softverske obrade velike količine dohvaćenih informacija u realnom vremenu. [2]

Strojni vid isključuje komponentu fizičkog kontakta s predmetom, koja je prisutna kod upotrebe ljudskog vida, smanjujući na taj način mogućnost oštećivanja predmeta te njegovo proizvodno vrijeme. Također, veća je i sigurnost radnika koji sudjeluju u proizvodnom procesu jer u njega nisu izravno uključeni. [3]

Strojni se vid primjenjuje u tehnologijama kontrole kvalitete, prepoznavanja, mjerenja, upravljanja, nadzora, kao i u mnogim drugima.



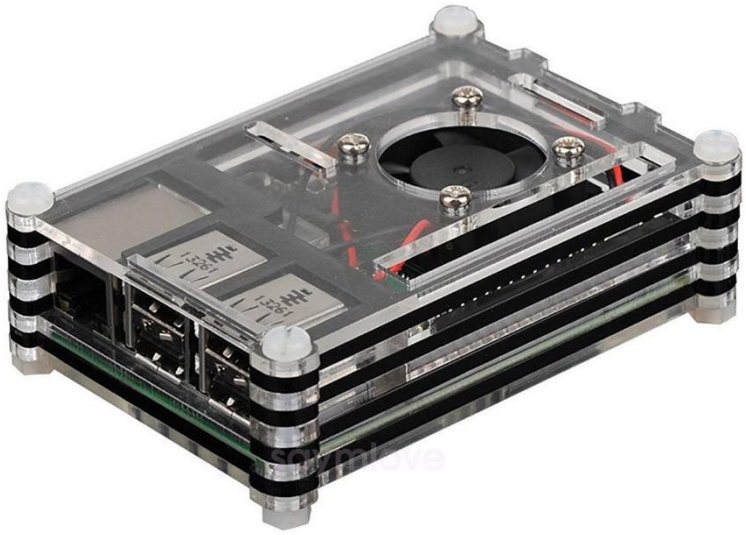
Slika 6. Primjena strojnog vida za određivanje gustoće prometa na određenom dijelu ceste

2. PROJEKTIRANJE VIZIJSKOG SUSTAVA

U ovom radu, vizijski je sustav temeljen na upravljačkoj jedinici Raspberry Pi Model 3 B+ zatvorenoj u plastično kućište radi zaštite od prašine te sa spojenim ventilatorom za aktivno hlađenje procesora. Na upravljačku je jedinicu preko USB konektora spojena web kamera, a koja se preko njega napaja i šalje snimljene video zapise upravljačkoj jedinici. Video zapisi se zatim procesiraju i u realnom vremenu prikazuju preko zaslona računala koje je s upravljačkom jedinicom spojeno SSH protokolom preko ethernet kabla. Na upravljačku je jedinicu preko GPIO (eng. General Purpose Input/Output) pinova spojen servo motor na koji je postavljena web kamera, a koji djeluje na temelju centriranja lokaliziranog i praćenog objekta u središte slike po horizontalnoj osi, također u realnom vremenu. Kriterij lokalizacije i praćenja objekta je da je objekt trenutno najveće površine na slici za određeni algoritam.

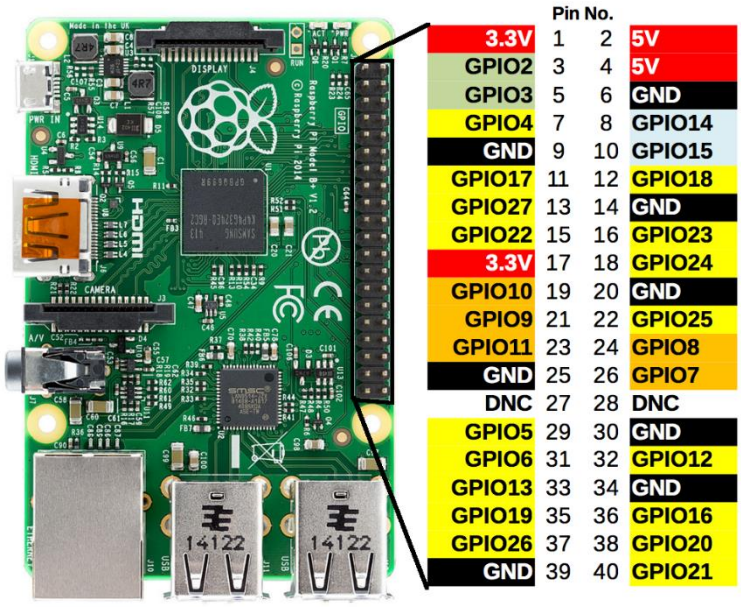
2.1. Upravljačka jedinica Raspberry Pi Model 3 B+

Raspberry Pi Model 3 B+ je mikroračunalo izvedeno na jednoj tiskanoj pločici razvijeno radi poticanja računalnih znanosti u nerazvijenim i zemljama u razvoju. Odabrano je zbog svoje široke dostupnosti i visoke procesorske moći (64-bitni četverojezgeni procesor frekvencije 1.4 GHz) u odnosu na druga mikroračunala, a koristi Raspbian operativni sustav, baziran na Debianu, koji je besplatan. Pruža vrlo dobru programsku podršku za programski jezik Python, koji je korišten u radu, a čiji su detaljni razlozi korištenja opisani u nastavku.



Slika 7. Raspberry Pi Model 3 B+ u kućištu sa spojenim ventilatorom

Raspberry Pi Model 3 B+ sadrži četrdeset GPIO pinova, koji se dijele na pinove s konstantnim naponom od 3.3 V, s konstantnim naponom od 5.0 V, za uzemljenje (eng. Ground - GND), s upravljivim naponom te na pinove na koje se ne spaja ništa (eng. Do Not Connect – DNC). Na sljedećoj je slici prikazan raspored GPIO pinova na upravljačkoj jedinici.



Slika 8. Raspored GPIO pinova na Raspberry Pi Model 3 B+

2.2. Servo motor

Servo motor je elektromotor koji prema primljenom upravljačkom signalu manje snage (mehaničkom, pneumatskom, hidrauličkom ili električnom) razvijanjem odgovarajućeg momenta, odnosno sile zauzima određeni zakretni položaj ako je rotacijski, odnosno određeni linearni položaj ako je linearni, koristeći pritom više snage. Pokretanje i zaustavljanje servo motora upravljačkim signalom odvija se preko regulacijskog kruga, koji je s izvršnim dijelom servo motora povezan mjernim uređajem koji preko povratne veze djeluje na dobiveni položaj. [6]

Pošto se radi samog snimanja video zapisa treba koristiti upravljačka jedinica koja je neophodna za upravljanje rotacijskim servo motorom, a i zbog visoke preciznosti rotacijskog servo motora te potrebe za samo malim momentom, u obzir se nisu uzimale druge vrste elektromotora, kao što su istosmjerni, koračni ili linearni servo motor.

Odabran je rotacijski servo motor Hitec HS-425BB, prikazan na sljedećoj slici, a njegove spefikacije su dane u sljedećoj tablici.

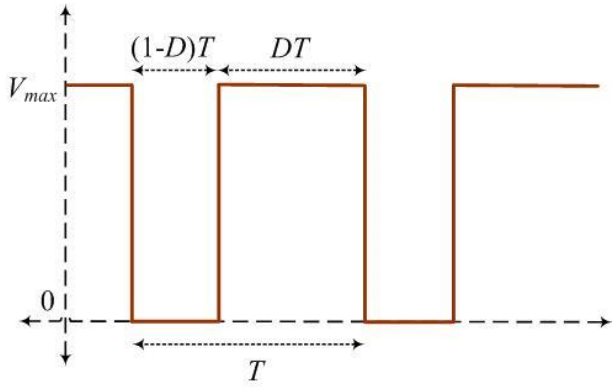
Slika 9. Rotacijski servo motor Hitec HS-425BB

Tablica 1. Specifikacije rotacijskog servo motora Hitec HS-425BB [7]

Masa	45.5 g
Gabaritne dimenzije	40.4 x 19.6 x 36.6 mm
Maksimalni kut zakreta	188°
Radni napon	4.8 V – 6.0 V
Brzina zakreta pri 4.8 V	0.286°/ms
Brzina zakreta pri 6.0 V	0.375°/ms
Izlazni moment pri 4.8 V	324 Nmm
Izlazni moment pri 6.0 V	402 Nmm
Radna električna struja pri 4.8 V	150 mA
Radna električna struja pri 6.0 V	180 mA
Napon upravljačkog signala	3.0 V – 5.0 V
Trajanje upravljačkog signala u periodi	0.553 ms – 2.520 ms
Smjer rotacije pri povećanju upravljačkog signala	u smjeru kazaljke na satu
Radna temperatura	-20°C – 60°C

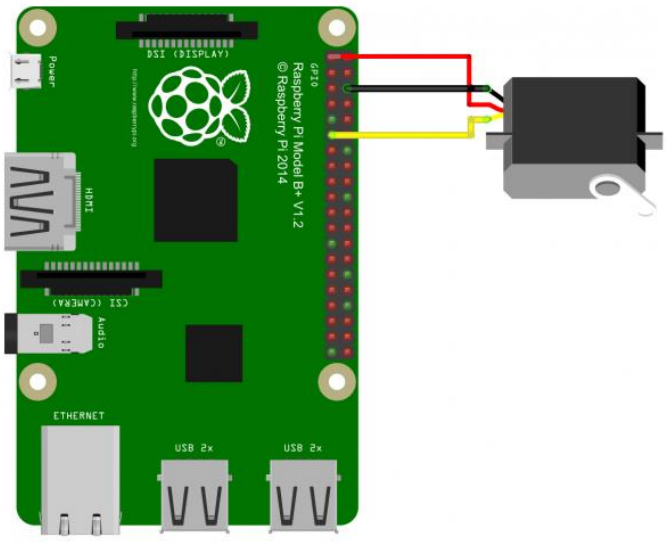
Odabrani je motor, kao i svaki drugi servo motor, upravljan pulsno-širinskom modulacijom (eng. Pulsewidth Modulation – PWM). To je način pretvorbe analognog signala u digitalni, pri

čemu se u svakoj periodi (T) od 20 ms generira pravokutni signal određenog napona (V_{max}) i trajanja (DT), čiji iznos određuje kut zakreta motora.. Za odabrani je motor taj napon 3.0 – 5.0 V, a trajanje od 0.553 ms, kada motor zauzima kut zakreta od 0°, pa do 2.520 ms, kada motor zauzima kut zakreta od 188°.



Slika 8. Princip pulsno-širinske modulacije

Servo motor sadrži tri izvoda: izvod za radni napon (crvena žica), izvod za upravljački napon (žutra žica) te izvod za uzemljenje (crna žica). Kako na Raspberry Pi-ju postoje pinovi s konstantnim naponom od 3.3 V i od 5.0 V te iako je radni napon odabranog servo motora 4.8 V – 6.0 V, pripadni izvod je zbog zaštite upravljačke jedinice i potrebe za samo malim momentom spojen na 3.3 V. Izvod za uzemljenje je spojen na pin za uzemljenje, a izvod za upravljački napon na pin s upravljivim naponom. Shema spajanja dana je na sljedećoj slici.



Slika 9. Shema spajanja servo motora i Raspberry Pi-ja

2.3. Web kamera

Za snimanje video zapisa odabrana je CANYON CNE-CWC1 web kamera s pokretljivim zglobom za namještanje položaja, maksimalne rezolucije slike 1600 x 1200 piksela te maksimalne rezolucije videa 640 x 480 piksela pri maksimalnoj brzini snimanja videa od 30 slika po sekundi. Na upravljačku je jedinicu spojena USB konektorom, preko kojeg se napaja i šalje snimljene video zapise upravljačkoj jedinici. Odabrana web kamera prikazana je na sljedećoj slici.

Slika 10. Web kamera CANYON CNE-CWC1

2.4. Komunikacija između upravljačke jedinice i računala

Komunikacija između upravljačke jedinice i računala odvija se SSH (eng. Secure Shell) protokolom preko ethernet kabla. SSH protokol je protokol za sigurnu komunikaciju između udaljenih odredišta u nesigurnoj računalnoj mreži, pri čemu je komunikacija podijeljena u tri sloja: transportni, autentifikacijski i spojni. SSH protokol se koristi za uspostavu grafičkog sučelja na računalu radi udaljenog upravljanja upravljačkom jedinicom, koristeći pritom ulazne jedinice računala (miš, tipkovnicu i zaslon). [8]

2.5. Programski jezik Python

Python je visoki interpreterski programski jezik čija je prva inačica je izdana 1990. godine te je ubrzo postao široko upotrebljavan. Ne donosi revolucionarne značajke u programiranju, već na optimalan način ujedinjuje najbolja načela rada drugih programskih jezika. Razvijen je s

naglaskom na čitljivost koda, tako da programeru omogućuje veću posvećenost problemu nego sintaksi. Otvorenog je koda, besplatan za akademske i komercijalne svrhe, s vrlo dobrom programskom potporom te sa mnoštvom knjižnica širokog spektra, što mu omogućava primjenu u raznim područjima.

Osim standardnih tipova podataka kao što su cijeli brojevi, decimalni brojevi, znakovni nizovi (eng. String) i logički tipovi, ima ugrađene i tipove podataka visoke razine, kao što su liste, n-terci i rječnici. Python je objekto orijentiran programski jezik, što znači da je usmjeren na izradu skupa objekata koji sadrže određene atribute te koji izmjenjuju podatke između sebe.

Nedostatak mu je što je izvršavanje kodova pisanih u njemu sporo, dosta sporije nego kod, primjerice, programskih jezika C i C++. Razlog tome je što je Python interpreterski jezik, dok su C i C++ kompajlerski. Razlike između interpreterski i kompajlerskih jezika su dane u sljedećoj tablici.

Tablica 2. Razlike između interpreterskih i kompajlerskih jezika [9]

interpreterski jezici	kompajlerski jezici
izvršavaju liniju po liniju izvornog koda bez njegovog prethodnog prevođenja	prevode cijeli izvorni kod u strojni kod te ga zatim iz strojnog koda izvršavaju odjednom
brzo analiziraju izvorni kod, ali ga dugo izvršavaju	dugo analiziraju izvorni kod, ali ga brzo izvršavaju
ne stvaraju strojni kod pa koriste manje memorije	stvaraju strojni kod pa su koriste više memorije
ako postoji greška u kodu, program staje odmah kod izvršavanja te linije koda pa je otklanjanje grešaka lako	ako postoji greška u kodu, program staje tek nakon pretvaranja cijelog izvornog koda u strojni

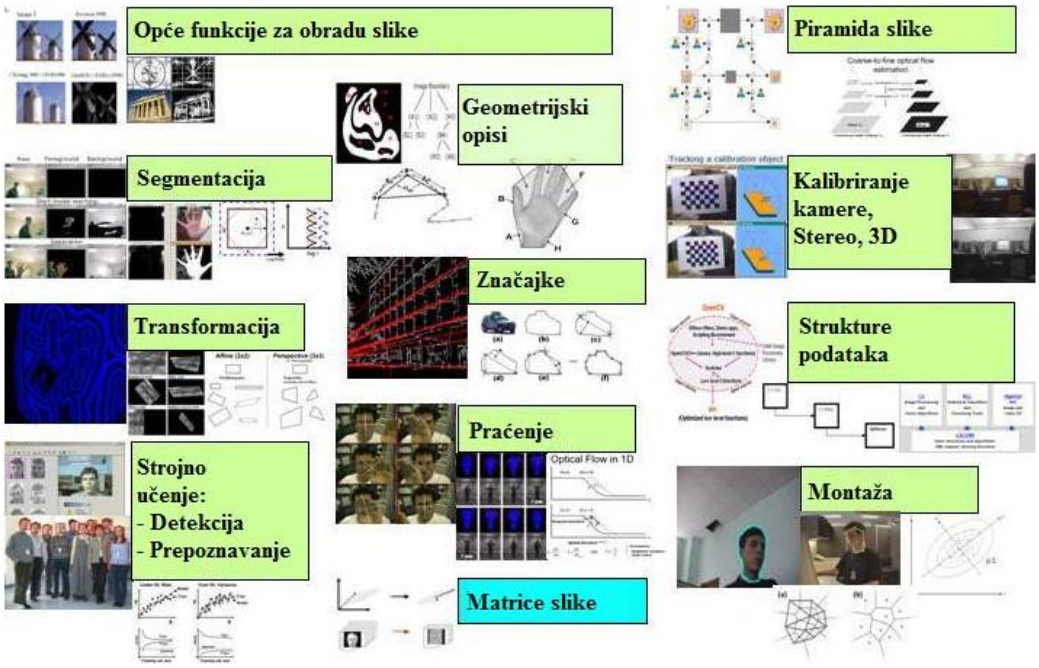
Problem sporog izvršavanja kodova pisanih u Python-u može se riješiti korištenjem programskog okruženja Cython, koje kod pisan u Pythonu prevodi u onaj pisan u C-u te ga dalje izvršava kao da je pisan u C-u.

Razlozi odabira Pythona, a ne C++-a, su to što za njega Raspberry Pi pruža puno bolju programsku podršku te što je u njemu jednostavnije pisati kodove, a to se pogotovo odnosi na izradu grafičkog sučelja.

2.6. OpenCV knjižnica otvorenog koda

OpenCV (eng. Open Source Computer Vision) je knjižnica otvorenog koda, besplatna za akademske i komercijalne svrhe, a čija je prva inačica izdana 2000. godine. Sadrži sučelja za rad u programskim jezicima C++, Python, MATLAB te Java. Podržava Windows, Linux, Android te još neke operativne sustave. Napisana je u C++ programskom jeziku te je za njega izvorno i optimizirana.

Razvijena je s ciljem učinkovite obrade slika u realnom vremenu. Danas sadrži više od 2500 optimiziranih algoritama u području računalnog vida i strojnog učenja, a koji obuhvaćaju opće funkcije obrade slike, segmentaciju dijelova slike, prepoznavanje rubova objekata, prepoznavanje određenih uzoraka na slikama, prepoznavanje pozadine slike, detekciju i identifikaciju objekata i lica, praćenje objekata i lica, stvaranje panoramskih slika, stvaranje 3D modela objekata, kalibriranje kamere te mnoge druge. Neke od navedenih te nekolicima drugih mogućnosti OpenCV knjižnice prikazane su na sljedećoj slici. [10]



Slika 11. Pregled najvažnijih mogućnosti OpenCV knjižnice

OpenCV knjižnica broji više od 14 milijuna preuzimanja, što pokazuje njenu široku upotrebljenost u razvoju aplikacija. Koriste je mnoge poznate tvrtke, poput Googlea, Microsofta, Intela, Sonyja te Honde.

3. PROGRAMSKO RJEŠENJE ZA LOKALIZACIJU I PRAĆENJE OBJEKATA

Lokalizacija određenih objekata provodi se prema jednoj od četiri metode:

- određivanju raspona boja iz HSV prostora boja
- unatražnoj projekciji podataka (eng. Backprojection)
- određivanju apsolutnih razlika između trenutne i prethodne slike
- određivanju gustog optičkog toka slike (eng. Dense Optical Flow).

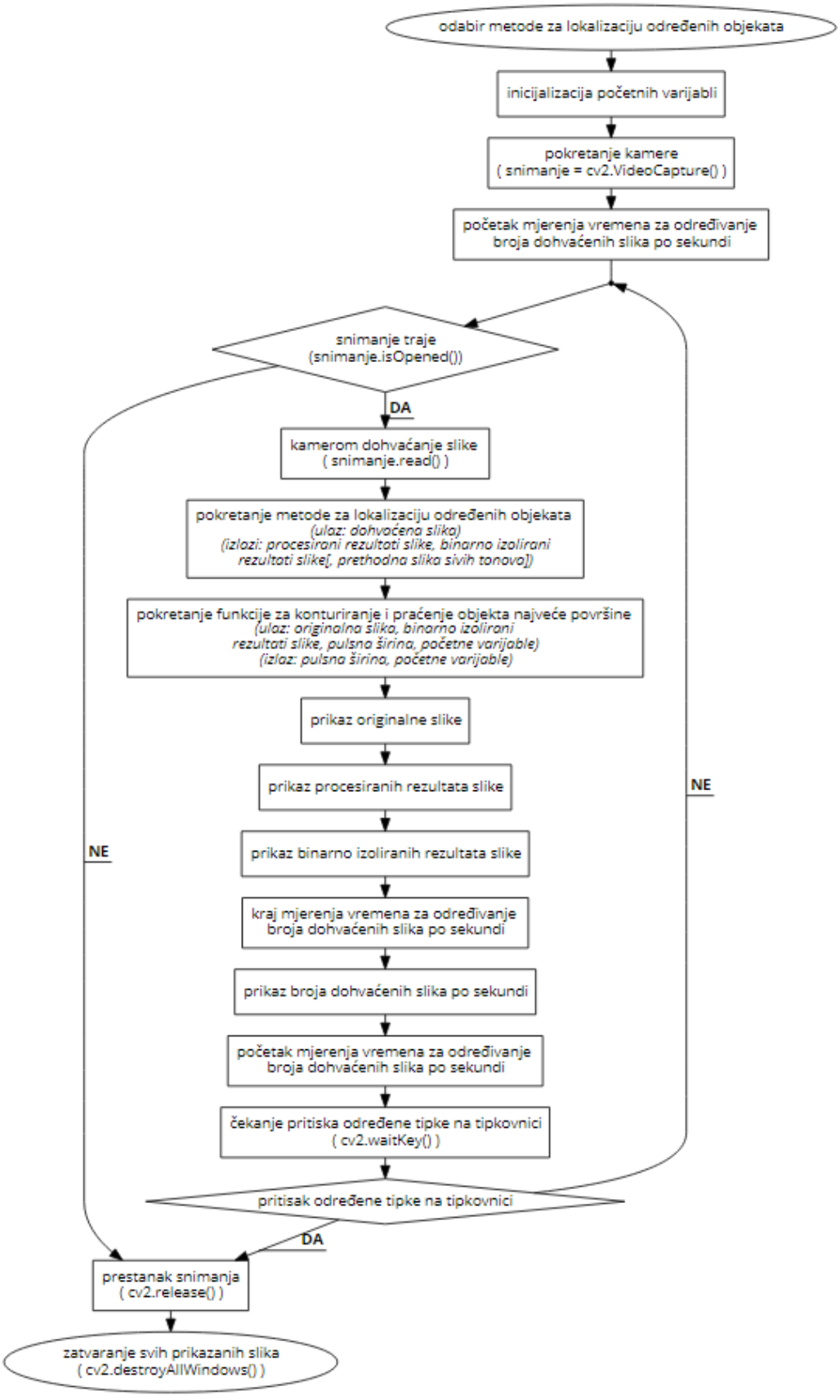
Zbog fizikalnog ograničenja da je moguće pratiti samo jedan objekt, od lokaliziranih zadovoljavajućih objekata se odabire samo onaj trenutno najveće površine na slici te ako je on stabilne površine i položaja na slici, određuje se njegovo središte, koje se pomoću servo motora kontinuirano pokušava centrirati u središte slike po horizontalnoj osi.

Pošto se potpuni algoritam između metoda razlikuje samo prema tome kojom metodom se lokaliziraju određeni objekti, analiza je izvedena tako da je najprije objašnjen potpuni algoritam, a tek onda same metode za lokalizaciju određenih objekata, algoritam odabira objekta trenutno najveće površine na slici i njegovog konturiranja te algoritam praćenja navedenog objekta.

3.1. Potpuni algoritam

Kod izvođenja potpunog algoritma se najprije odabere metoda za lokalizaciju određenih objekata. Zatim se inicijaliziraju početne varijable, izraziom *snimanje = cv2.VideoCapture()* se pokrene snimanje te se počne mjeriti vrijeme za određivanje broja dohvaćenih slika po sekundi (eng. Frames per Second – FPS). Kada se snimanje izvršava, što se provjerava funkcijom *snimanje.isOpened()*, izvršava se petlja u kojoj se odvija sljedeće: dohvaća se slika, funkcijom *snimanje.read()*, pokrene se metoda za lokalizaciju određenih objekata, pokrene se funkcija za konturiranje i praćenje objekta najveće površine, prikažu se metodom obrađene slike, završi se mjerenje vremena, prikaže se broj dohvaćenih slika po sekundi, nakon čega počinje novo mjerenje vremena te se pomoću funkcije *cv2.waitKey()* kratko vrijeme čeka pritisak određene tipke na tipkovnici. Ako je tipka pritisnuta, petlja se prekida, snimanje prestaje pomoću funkcije *cv2.release()* te se pomoću funkcije *cv2.destroyAllWindows()* zatvaraju sve prikazane slike.

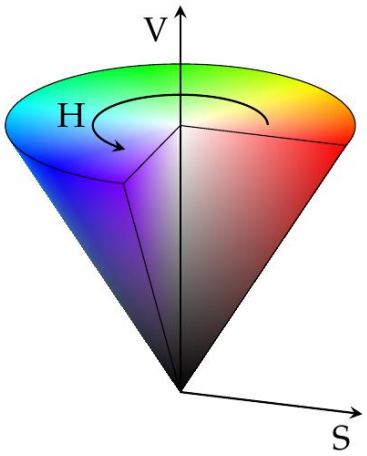
Dijagram toka potpunog algoritma:



3.2. Metode za lokalizaciju određenih objekata

3.2.1. Metoda određivanja raspona boja iz HSV prostora boja

HSV prostor boja koristi nijansu (eng. Hue), zasićenje (eng. Saturation) i vrijednost (eng. Value) za opisivanje pojedine boje. Razvijen je s ciljem da bi što bolje predstavljao način na koji ljudsko oko percipira boje.



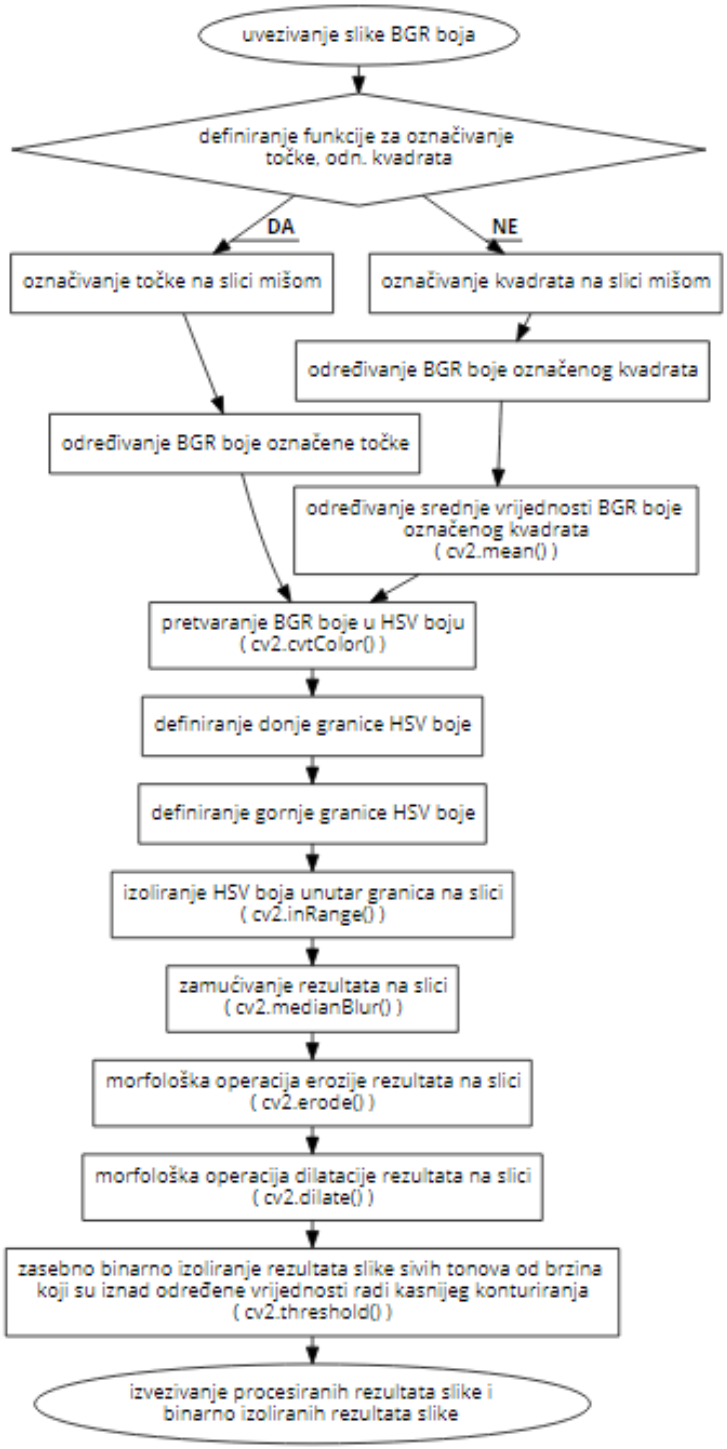
Slika 12. HSV prostor boja

OpenCV se u kod napisan u Pythonu uvezuje izrazom *import* cv2. Ne koristi RGB prostor boja, već BGR, tj. kanali za crvenu i plavu boju su mu zamijenjeni. Pošto se u RGB i BGR prostoru boja boje međusobno sličnih nijansi obično razlikuju u sva tri kanala, za njihovo je izoliranje logičnije koristiti HSV prostor boja. Boja se u OpenCV-u između raznih prostora boja pretvara funkcijom *cv2.cvtColor()*. Zasićenje boje ovisi o udjelu sive koji ona sadrži, a vrijednost boje o osvjetljenju slike. Kada se objekt pomiče u odnosu na kameru, mijenja se osvjetljenje slike, pri čemu se vrijednost boje objekta, no indirektno se mijenja i zasićenje njegove boje. Nijansa boje pokazuje neznatnu promjenu, no ona svakako treba biti uzeta u obzir. Kod ove se metode, kao i kod ostalih, najprije uveze kamerom dohvaćena slika u BGR prostoru boja. Da bi se izolirale određene međusobno slične boje, postavljaju se donja i gornja granica HSV boje u odnosu na srednju HSV boju objekta. Područje između granica nijansi boja je vrlo usko, dok su područja između granica zasićenja i vrijednosti boja vrlo široka. Boje se u OpenCV-u izoliraju funkcijom *cv2.inRange()*. Da bi izolirane boje činile likove čije su površine što vjernije površinama likova sa uvezene slike, rezultati na slici se zamućuju, funkcijom *cv2.medianBlur()*, te se na njih primjenjuju

morfološke operacije erozije, funkcijom *cv2.erode()*, i nakon nje dilatacije, funkcijom *cv2.dilate()*. Zamućivanje se vrši radi smanjenja šumova koji se javljaju na slici, a funkcionira na način da se iznos neke veličine svakog piksela zamjenjuje srednjim iznosom iste veličine njegovih okolnih piksela. Erozija i dilatacija se koriste za micanje individualnih objekata male površine koji se javljaju na slikama. Kod svijetlih likova na slici funkcioniraju tako da erozija smanjuje njihovu debljinu, a dilatacija je povećava, dok kod tamnih funkcioniraju obrnuto. Pošto se u korištenim metodama fokusira na svijetle likove, u svakoj metodi se najprije vrši erozija, a zatim dilatacija. Zatim se funkcijom *cv2.threshold()* birnarno izoliraju rezultati koji su iznad određene vrijednosti radi kasnijeg konturiranja. Binarno izoliranje funkcionira tako da iznosima neke veličine iznad određene vrijednosti dodjeljuje jedan iznos, a ispod određene vrijednosti drugi iznos. Naposljetku se u glavnu petlju algoritma izvezu procesirani rezultati slike i binarno izolirani rezultati slike, koji će se koristiti za konturiranje.

Slika 13. Originalni lik, dilatirani lik i erodirani lik

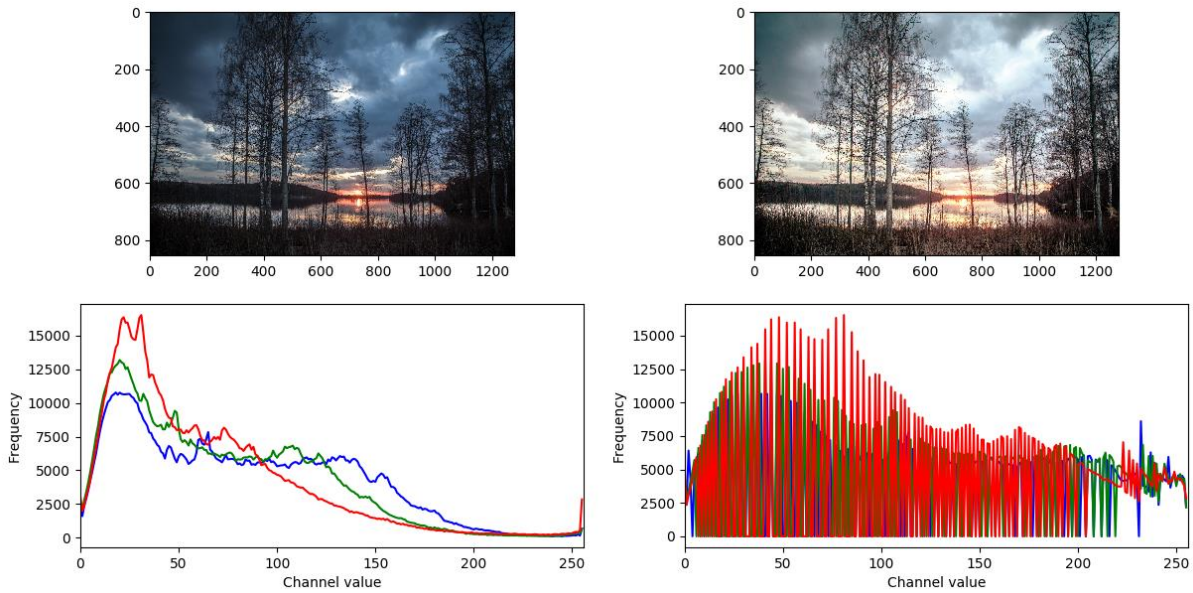
Dijagram toka lokaliziranja objekata određivanjem raspona boja iz HSV prostora boja:



3.2.2. Metoda unatražne projekcije podataka

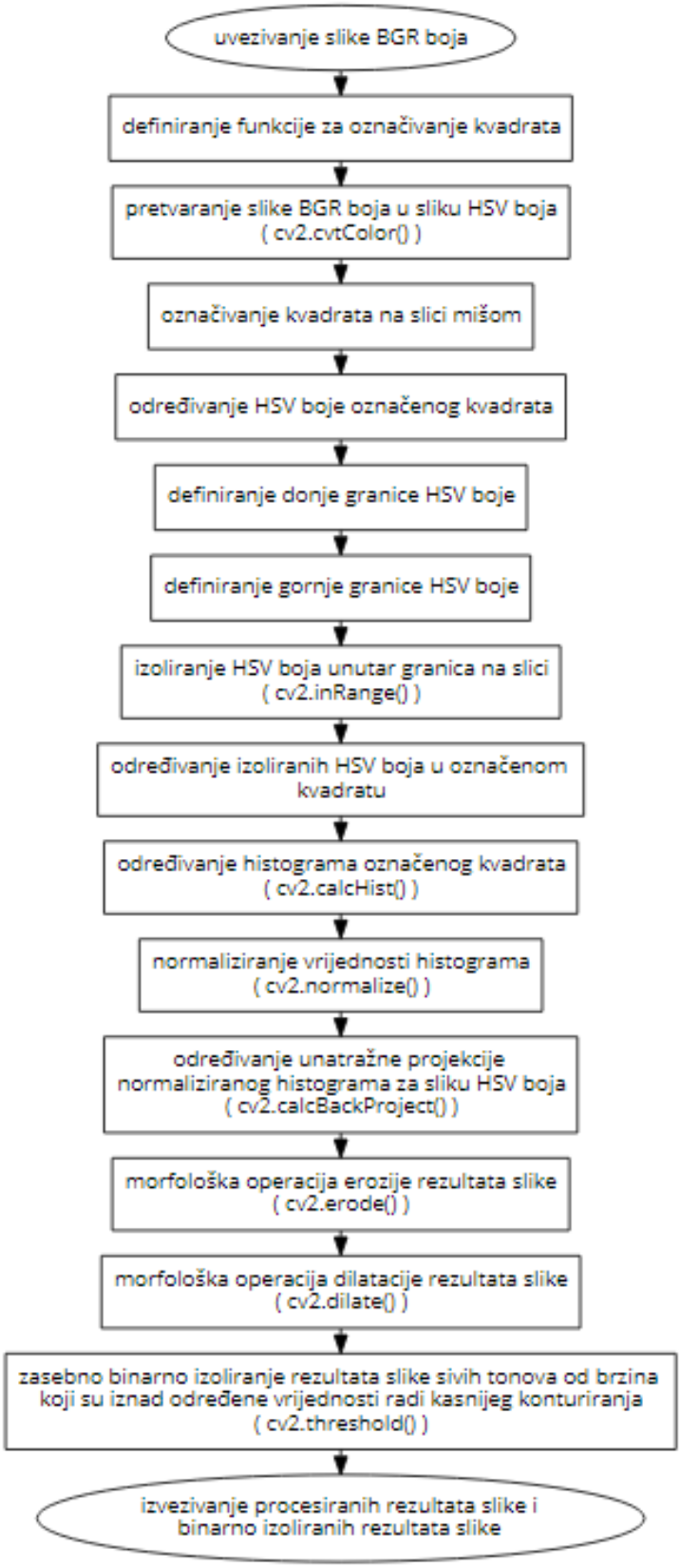
Navedena metoda koristi određivanje histograma slike koji su grafička reprezentacija razdiobe iznosa određenih kanala boje slike na pojedinim dijelovima slike koji mogu biti sastavljeni od

jednog ili više piksela po širini i jednog ili više piksela po visini. OpenCV podržava jednodimenzionalne i dvodimenzionalne histograme, tj. nije moguće koristiti tri kanala boje slike, već maksimalno dva, pri čemu je zbog u prošloj metodi navedenih razloga, najbolje koristiti nijansu i zasićenje boje iz HSV prostora boja. Najprije se uveze kamerom dohvaćena slika u BGR prostoru boja, koja se zatim funkcijom *cv2.cvtColor()* pretvara u HSV prostor boja. Da bi se eliminirale boje pri slabom osvjetljenju, postavljaju se donja i gornja granica HSV boje, pri čemu donja granica ima najnižu moguću nijansu te zasićenje i vrijednost dovoljno visoke da se eliminiraju navedene boje, a gornja granica ima najvišu moguću svaku veličinu, te se primjenjuje funkcija *cv2.inRange()*. Nakon toga se funkcijom *cv2.calcHist()* određuje histogram označenog dijela slike čiji se objekt želi pratiti te se taj histogram na standardne vrijednosti svodi normalizacijom, pomoću funkcije *cv2.normalize()*. Normalizacijom histograma se poboljšava kontrast same slike. Sljedeći je korak korištenje unatražne projekcije normaliziranog histograma, čiji je cilj pronaći područje na slici koje odgovara razdiobi iznosa određenih kanala boje označenog dijela slike, a vrši se pomoću funkcije *cv2.calcBackProject()*. Unatražna projekcija kao rezultat daje vjerojatnost da određeno područje odgovara toj razdiobi. Na rezultate na slici se zatim primjenjuju morfološke operacije erozije, funkcijom *cv2.erode()*, i dilatacije, funkcijom *cv2.dilate()*, čija je svrha jednaka kao i u prethodnoj metodi. Funkcijom *cv2.threshold()* se, na isti način kao i u prethodnoj metodi, birnarno izoliraju rezultati koji su iznad određene vrijednosti radi kasnijeg konturiranja. Na kraju se u glavnu petlju algoritma izvezu procesirani rezultati slike i binarno izolirani rezultati slike, koji će se koristiti za konturiranje.



Slika 14. Originalna slika i njen histogram te normalizirana slika i njen histogram

Dijagram toka lokaliziranja objekata unatražnom projekcijom podataka:

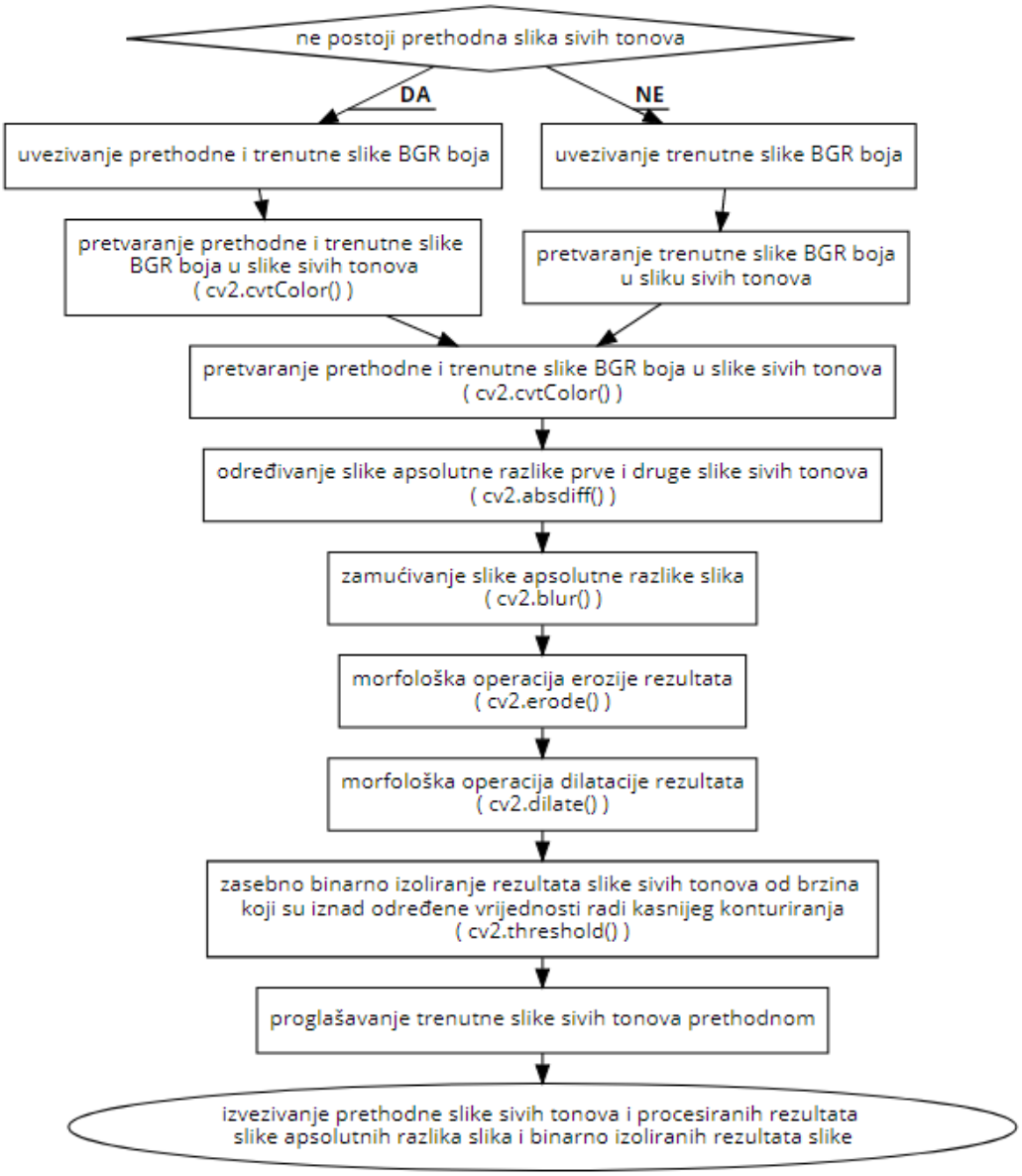


3.2.3. Metoda određivanja apsolutnih razlika između trenutne i prethodne slike

Navedena metoda koristi trenutnu i prethodnu sliku dohvaćenu kamerom u BGR prostoru boja, koje se zatim pretvorene u prostor boja sivih tonova funkcijom *cv2.cvtColor()* te se odredi njihova međusobna apsolutna razlika funkcijom *cv2.absdiff()*. Apsolutna razlika slika odgovara onim dijelovima slike koji su se pomaknuli u odnosu na prethodnu sliku, pri čemu se kod praćenja javlja problem pomaka velikog dijela slike zbog pomicanja same kamere uslijed djelovanja servo motora. To je riješeno na način da se lokalizirani objekti ne konturiraju nekoliko slika nakon što se kamera počne pomicati. Algoritam se nastavlja zamućivanjem rezultata na slici funkcijom *cv2.blur()* te primjenom morfoloških operacija erozije, funkcijom *cv2.erode()*, i nakon nje dilatacije, funkcijom *cv2.dilate()*, a čija je čija je svrha jednaka kao i u prethodnim metodama. Funkcijom *cv2.threshold()* se, na isti način kao i u prethodnim metodam, birnarno izoliraju rezultati koji su iznad određene vrijednosti radi kasnijeg konturiranja. Trenutna slika sivih tonova se proglasi prethodnom da bi se sa onom koja će biti uvezena sljedeća mogla koristiti kod određivanja nove apsolutne razlike slika. Na kraju se u glavnu petlju algoritma izvezu prethodna slika sivih tonova, procesirani rezultati slike i binarno izolirani rezultati slike, koji će se koristiti za konturiranje.

Slika 15. Princip djelovanja apsolutne razlike slika

Dijagram toka lokaliziranja objekata određivanjem apsolutnih razlika između trenutne i prethodne slike:



3.2.4. Metoda određivanja gustog optičkog toka slike

Navedena metoda koristi trenutnu i prethodnu sliku dohvaćenu kamerom u BGR prostoru boja, koje se zatim pretvorene u prostor boja sivih tonova funkcijom *cv2.cvtColor* te se odredi njihov gusti optički tok funkcijom *cv2.calcOpticalFlowFarneback()*. Gusti optički tok je termin za dvodimenzionalno polje vektora brzina pridruženih svakom pikselu slike sivih tonova na temelju njihove promjene položaja u odnosu na prethodnu sliku. Kod praćenja se, kao i u prethodnoj metodi, javlja problem pomaka velikog dijela slike zbog pomicanja same kamere

uslijed djelovanja servo motora. To je također riješeno na način da se lokalizirani objekti ne konturiraju nekoliko slika nakon što se kamera počne pomicati.

Optički tok pretpostavlja da se iznos kanala boje pojedinog objekta ne mijenja te da susjedni pikseli imaju slično gibanje. Temelji se na formuli da je stanje svakog piksela određeno njegovom horizonalnom i vertikalnom koordinatom na slici te vrenemom, što je izraženo funkcijom:

$$F(x, y, t) = F(x + dx, y + dy, t + dt),$$

pri čemu je $F(x, y, t)$ početno stanje nekog piksela, a $F(x + dx, y + dy, t + dt)$ završno. dx, dy i dt označavaju promjenu horizontalne i vertikalne koordinate na slici te vremena. Razvojem u Taylorov red se dobiva funkcija:

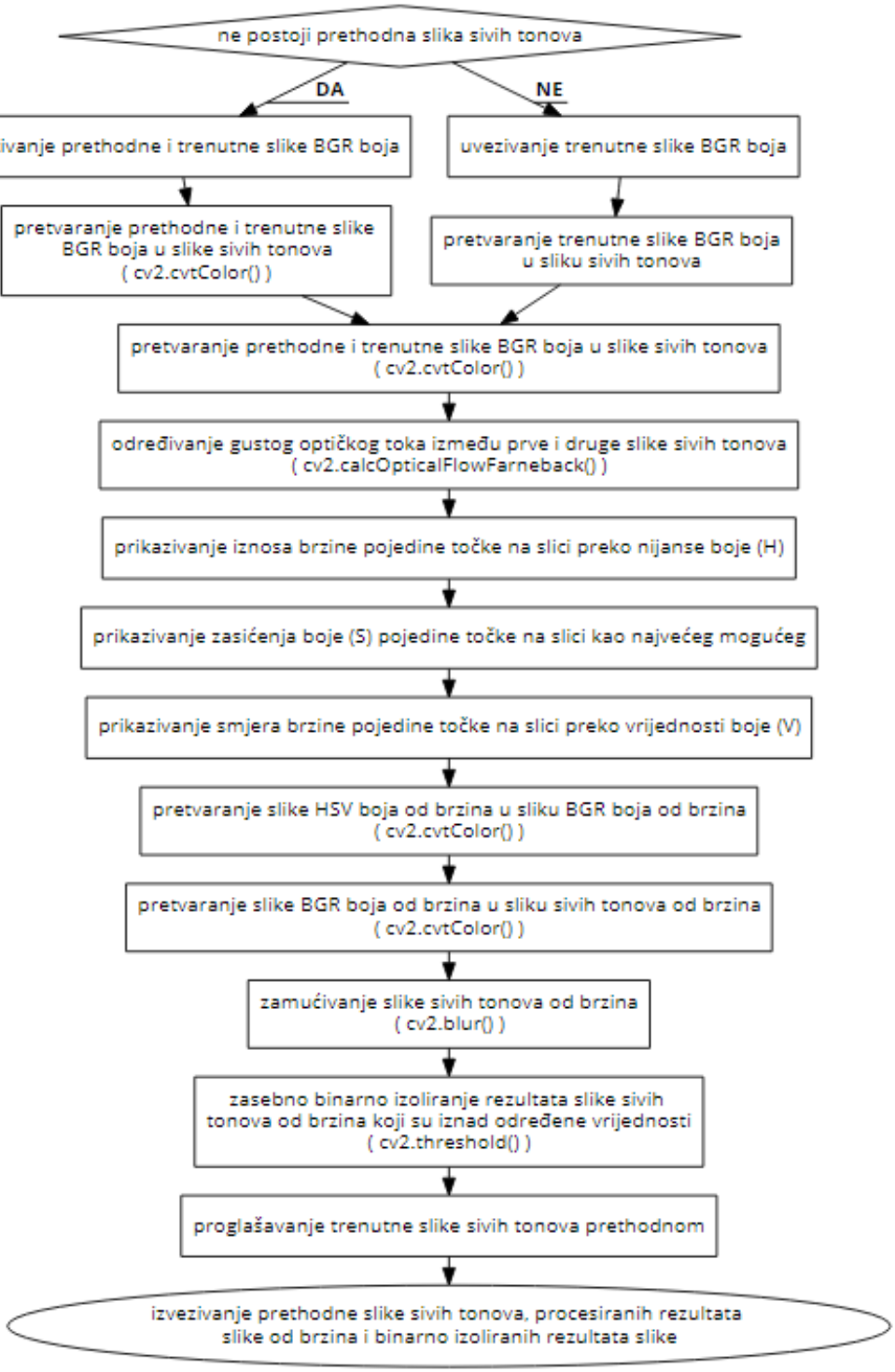
$$\frac{\partial f}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial f}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial f}{\partial t} = 0,$$

u kojoj su $f_x = \frac{\partial f}{\partial x}$, $f_y = \frac{\partial f}{\partial y}$ i $f_t = \frac{\partial f}{\partial t}$ gradijenti horizontalne i vertikalne koordinate na slici te vremena, a $u = \frac{dx}{dt}$ i $v = \frac{dy}{dt}$ iznosi brzine po horizontalnim i vertikalnim koordinatama na slici, koji su ujedno i jedine nepoznanice. Primjenom metode najmanjih kvadrata mogu se izračunati iz sljedeće matrične jednadžbe:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}{}^2 & \sum_i f_{x_i} \cdot f_{y_i} \\ \sum_i f_{x_i} \cdot f_{y_i} & \sum_i f_{y_i}{}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} \cdot f_{t_i} \\ -\sum_i f_{y_i} \cdot f_{t_i} \end{bmatrix}$$

Kada se za svaki piksel izračunaju iznosi brzina po horizontalnim i vertikalnim koordinatama na slici, oni se pretvore u vektore brzina čiji se iznos prikazuje preko nijanse boje, a smjer preko vrijednosti boje, dok se za zasićenje boje uzima najveći mogući iznos. Dobivena slika se iz HSV prostora boja funkcijom *cv2.cvtColor()* najprije pretvara u BGR prostor boja, a zatim u prostor boja sivih tonova. Nakon toga se dobiveni rezultati na slici zamućuju funkcijom *cv2.blur()*, čija je čija je svrha jednaka kao i u prethodnim metodama. Funkcijom *cv2.threshold()* se, na isti način kao i u prethodnim metodama, birnarno izoliraju rezultati koji su iznad određene vrijednosti radi kasnijeg konturiranja. Trenutna slika sivih tonova se proglasi prethodnom da bi se sa onom koja će biti uvezena sljedeća mogla koristiti kod određivanja novog gustog optičkog toka. Na kraju se u glavnu petlju algoritma izvezu prethodna slika sivih tonova, procesirani rezultati slike i binarno izolirani rezultati slike, koji će se koristiti za konturiranje.

Dijagram toka lokaliziranja objekata određivanjem gustog optičkog toka slike:

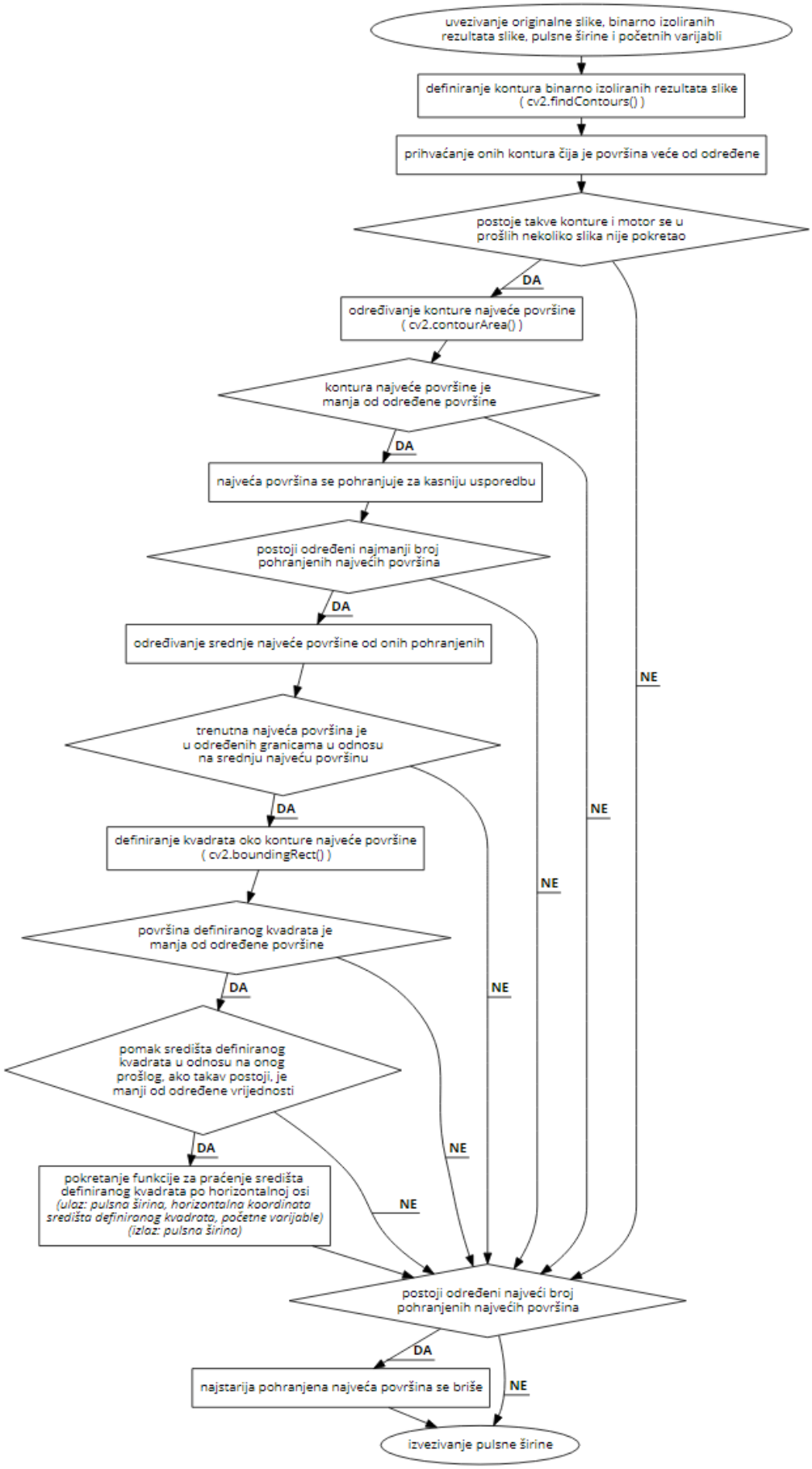


Slika 16. Slika s vektorima brzina prikazim pomoću linia, originalna slika i slika s vektorima brzina prikazanim u HSV sustavu boja

3.3. Algoritam odabira objekta trenutno najveće površine na slici i njegovog konturiranja

Navedeni algoritam funkcionira na principu prikazanom u sljedećem dijagramu toka, a koristi funkcije *cv2.findContours()* za definiranje kontura binarno izoliranih rezultata slike, *cv2.contourArea()* za određivanje površina kontura te *cv2.boundingRect()* za definiranje kvadrata opisanog oko konture.

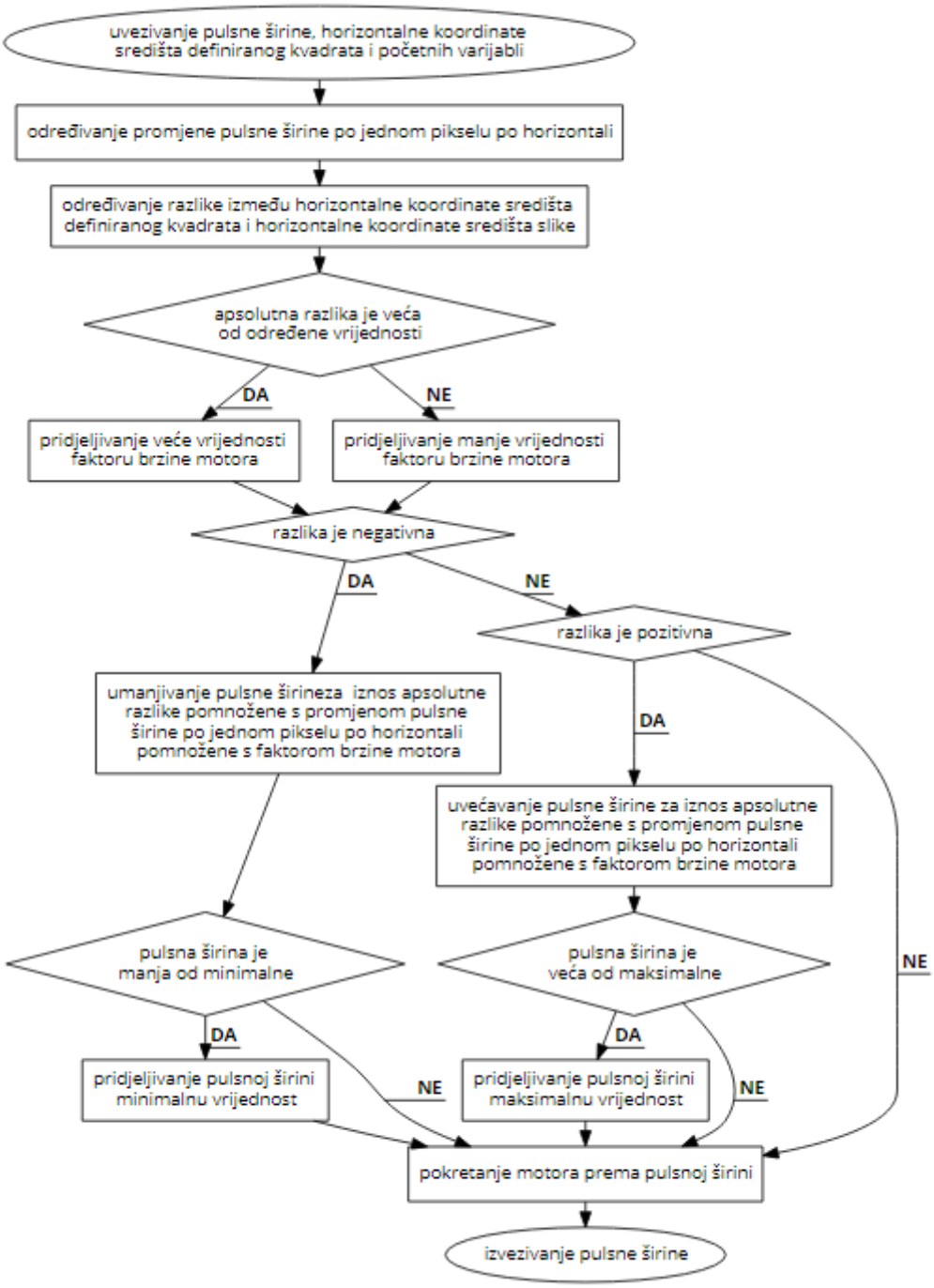
Dijagram toka navedenog algoritma:



3.4. Algoritam praćenja objekta trenutno najveće površine na slici

Praćenje funkcionira na principu prikazanom u sljedećem dijagramu toka, pri čemu se kut zakreta motora definira preko pulsne širine, koja se određuje iz algoritma. Za upravljanje motorom se koristi knjižnica pigpio, koja se u kod napisan u Pythonu uvezuje izrazom *import pigpio*. Postavljanje motora u određeni kut zakreta se vrši funkcijom *pigpio.pi().set_servo_pulsewidth()*.

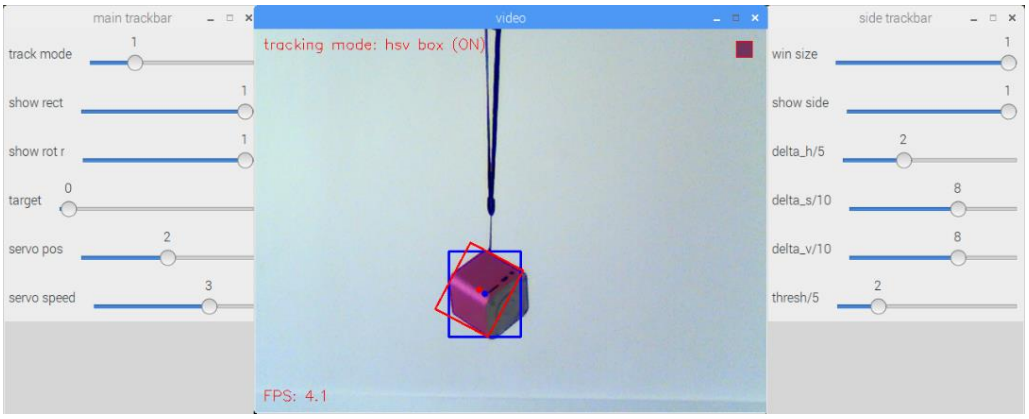
Dijagram toka praćenja objekta trenutno najveće površine na slici:



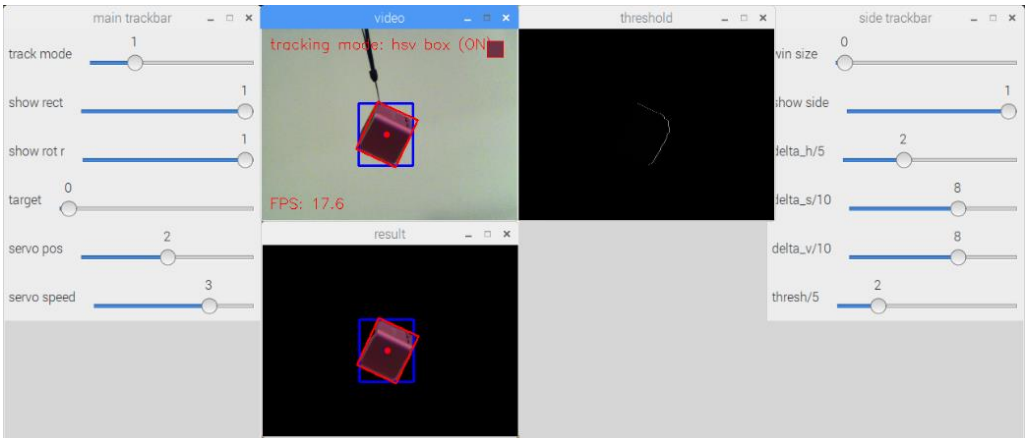
4. EVALUACIJA METODA ZA LOKALIZACIJU ODREĐENIH OBJEKATA I REZULTATI PRAĆENJA

4.1. Evaluacija metode određivanja raspona boja iz HSV prostora boja i praćenja

Predstavljena metoda daje odlične rezultate lokalizacije za objekte koji su boja dosta različitih od ostalih boja na slici. To uključuje jarke i matirane boje. Rezultalti su neiskoristivi za boje bijelih, sivih i crnih nijansi zbog iznosa njihovih veličina u HSV prostoru boja. Metoda pri rezoluciji videa od 640 x 480 piksela radi na jedva 4 do 5 slika po sekundi, a pri rezoluciji videa od 320 x 240 piksela na stabilnih 16 do 18 slika po sekundi. Praćenje je za dobro lokalizirane objekte vrlo robusno i stabilno.



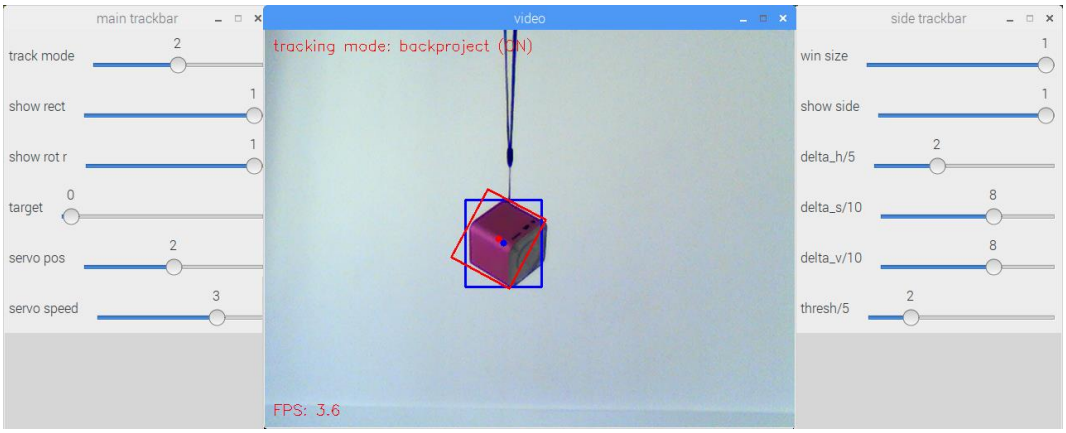
Slika 17. Rezultati metode određivanja raspona boja iz HSV prostora boja za rezoluciju videa od 640 x 480 piksela



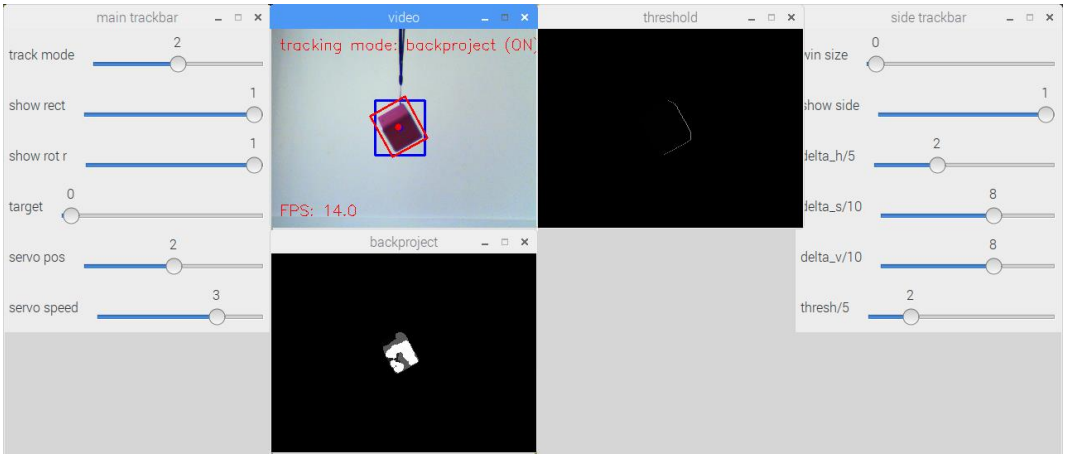
Slika 18. Rezultati metode određivanja raspona boja iz HSV prostora boja za rezoluciju videa od 320 x 240 piksela

4.2. Evaluacija metode unatražne projekcije podataka i praćenja

Ova metoda daje odlične rezultate lokalizacije za jednobojne objekte većine boja, dok za višebojne objekte ne daje iskoristive rezultate. Navedeno je uvjetovano potrebom da histogram označenog dijela objekta bude što izraženiji samo za uski raspon boja, da se ne lokaliziraju neželjeni objekti. Metoda pri rezoluciji videa od 640 x 480 piksela radi na 3 do 5 slika po sekundi, a pri rezoluciji videa od 320 x 240 piksela na stabilnih 14 do 18 slika po sekundi. Praćenje je za jednobojne objekte vrlo robusno i stabilno.



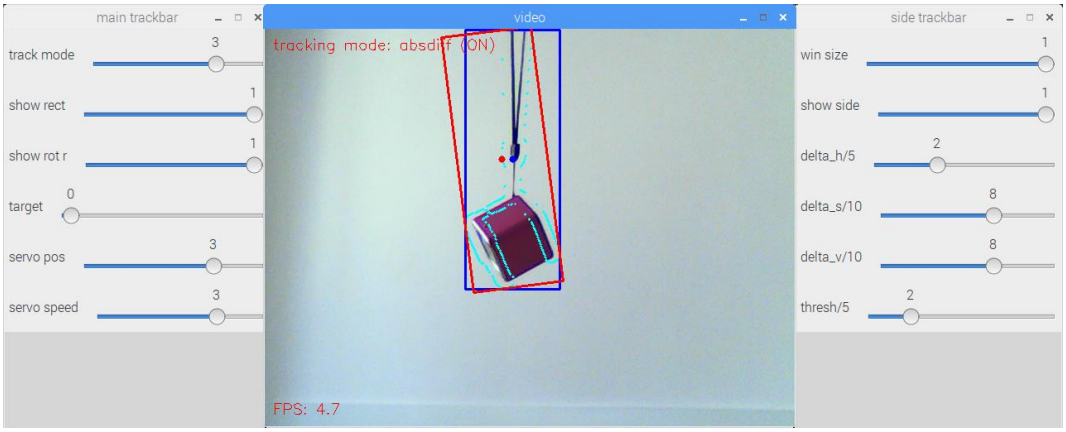
Slika 19. Rezultati metode unatražne projekcije podataka za rezoluciju videa od 640 x 480 piksela



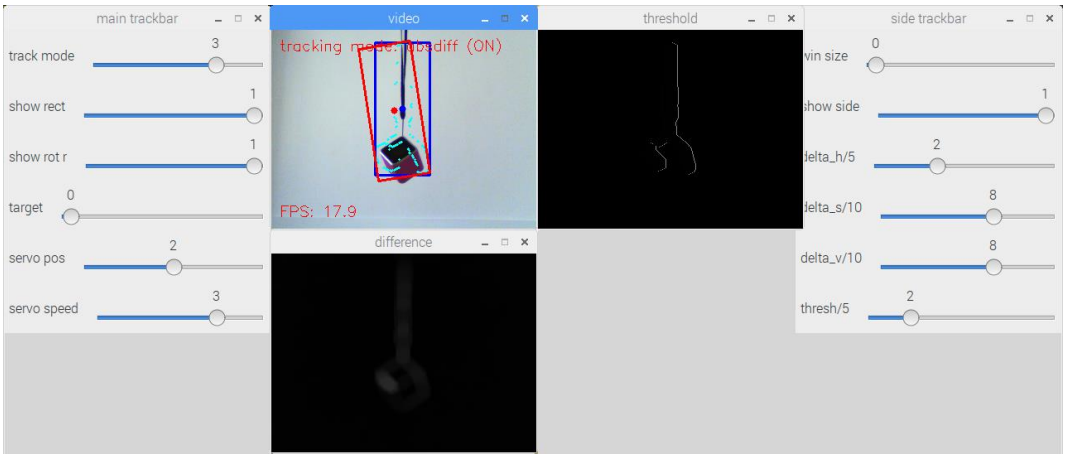
Slika 20. Rezultati metode unatražne projekcije podataka za rezoluciju videa od 320 x 240 piksela

4.3. Evaluacija metode određivanja apsolutnih razlika između trenutne i prethodne slike i praćenja

Predstavljena metoda daje vrlo dobre rezultate lokalizacije za bilo koje objekte koji se pomiču na slici, uz pretpostavku da u pozadini nema izoliranih objekata. Kada se u pozadini nalaze izolirani objekti, zbog pomicanja kamere se i oni miču, a informaciju o tom pomaku tada nije moguće otkloniti uvjetovanjem najveće prihvatljive površine jer im površina nije prevelika. Metoda pri rezoluciji videa od 640 x 480 piksela radi na 4 do 6 slika po sekundi, a pri rezoluciji videa od 320 x 240 piksela na stabilnih 16 do 20 slika po sekundi. Praćenje je vrlo stabilno te zadovoljivo robusno, a razlog tome je što se mijenjaju dijelovi objekta koji se gibaju, tj. ne giba se cijelo vrijeme cijeli objekt ili samo jedan njegov dio.



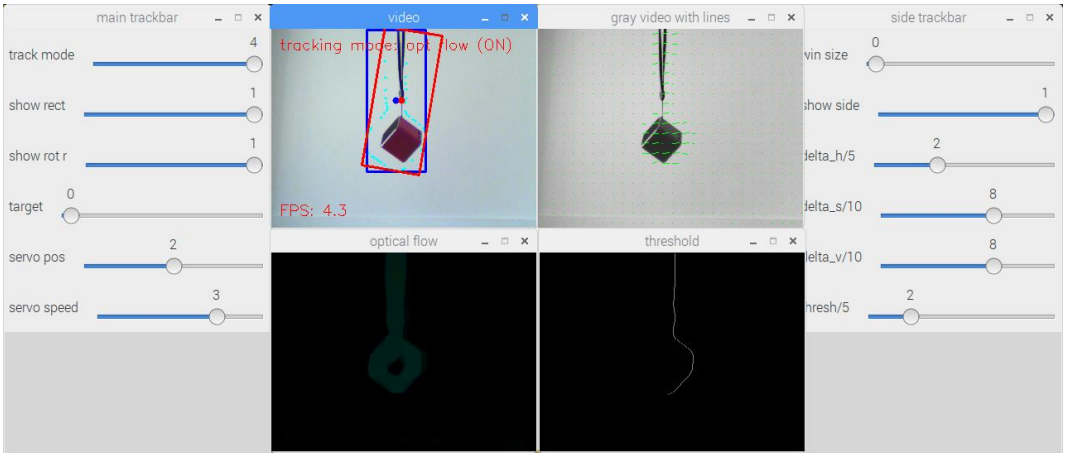
Slika 21. Rezultati metode određivanja apsolutnih razlika između trenutne i prehodne slike za rezoluciju videa od 640 x 480 piksela



Slika 22. Rezultati metode određivanja apsolutnih razlika između trenutne i prehodne slike za rezoluciju videa od 320 x 240 piksela

4.4. Evaluacija metode određivanja gustog optičkog toka slike i praćenja

Ova metoda ima potencijal dati odlične rezultate lokalizacije za bilo koje objekte koji se pomiču na slici, no uz upotrebu vlastite funkcije za određivanje gustog optičkog toka i uz korištenje bolje opreme, tj. veće brzine procesiranja slika. Realni dobiveni rezultati lokalizacije su iskoristivi za bilo koje objekte, no samo pri rezoluciji videa od 320 x 240 piksela. Kao i u prethodnoj metodi, kada se u pozadini nalaze izolirani objekti, zbog pomicanja kamere se i oni miču, a informaciju o tom pomaku tada nije moguće otkloniti uvjetovanjem najveće prihvatljive površine jer im površina nije prevelika. Metoda pri rezoluciji videa od 640 x 480 piksela radi na svega 1 do 2 slika po sekundi, što se ne može koristiti ni u koje svrhu, no pri rezoluciji videa od 320 x 240 piksela na 4 do 6 slika po sekundi, što je dovoljno za samo praćenje. Praćenje je stabilno onoliko koliko to oprema dopušta te zadovoljivo robusno, a razlog tome je što se mijenjaju dijelovi objekta koji se gibaju, tj. ne giba se cijelo vrijeme cijeli objekt ili samo jedan njegov dio.



Slika 23. Rezultati metode određivanja gustog optičkog toka slike za rezoluciju videa od 320 x 240 piksela

5. INTEGRACIJA DODATNIH ZNAČAJKI, MOGUĆA POBOLJŠANJA I MOGUĆA PRIMJENA

5.1. Integracija dodatnih značajki

Program je izveden sa kvaziparaleliziranjem dohvaćanja slike i njene obrade, bez čega bi broj slika po sekundi kod nekih metoda bio jedva dovoljan za samo praćenje, a kod nekih ni to. Napravljeno je sučelje koje omogućuje ručno podešavanje parametara kao što su raspon područja veličina dohvaćene slike, između granica pojedinih veličina HSV prostora boja, granica za binarno izoliranje rezultata slike, prikaz procesiranih slika, kut zakreta motora i brzina motora. Također, postoji mogućnost korištenja programa samo za lokalizaciju određenih objekata ili na upravljačkoj jedinici ili na računalu, bez praćenja pomoću servo motora.

5.2. Moguća poboljšanja

Tijekom rada uviđeno je da su za neke metode, zbog velikog broja operacija koje se trebaju izvršiti u realnom vremenu, potrebna brža izvršavanja kodova. Tome se može doskočiti obradom slika na grafičkim procesorima, koji imaju velik broj jezgri i mogu paralelno izvršavati velik broj relativno jednostavnih operacija, kakve su u ovom slučaju i prisutne. Sustavu bi dobro došla i bežična komunikacija sa računalom, da se može postavljati na mjesta na kojima pristup računalom nije moguć. Naposljetku, ako bi se sustav postavljao na mjesta sa kojih bi objekte bilo potrebno pratiti i po vertikalnoj osi, primjerice, na uglove prostorija ili zgrada, idealno bi bilo dodati još jedan servo motor. Algoritam za upravljanje tim motorom bi se od sadašnjeg algoritma za upravljanje motorom razlikovao u svega par detalja.

5.3. Moguća primjena

Projektiran je vizijski sustav moguće koristiti u razne svrhe. Idealnu bi primjenu mogao naći kao sigurnosna kamera koja lokalizira i prati određeni objekt, i to bez mnogo izmjena. S određenim izmjenama može poslužiti za brojanje objekata koji prođu kroz sceni ili za određivanje njihove brzine gibanja.

6. ZAKLJUČAK

Cilj je ovog završnog rada bio projektirati i integrirati robustan i stabilan sustav za lokalizaciju i praćenje određenih objekata te objasniti njegov način rada. Sustav je pokazao zadovoljavajuće rezultate te ga je moguće koristiti u razne svrhe. Kao takav može biti polažište za razvoj sustava za detekciju, identifikaciju i praćenje lica ili prethodno definiranih objekata. Spektar mogućnosti je vrlo širok te je ograničen jedino kreativnošću, znanjem i opremom.

Sve većim utjecajem automatizacije u industriji i sve većim zathjevima za kvalitetom proizvoda, vizijski sustavi se sve više primjenjuju i razvijaju. Primjenom vizijskih sustava može se postići znatno smanjenje troškova i povećanje fleksibilnosti sustava, ali je za njihovu uspješnu primjenu potrebno poznavati njihov način rada.

7. LITERATURA

[1] Owen-Hill, A.: *Robot Vision vs Computer Vision: What's the Difference?*, https://www.roboticstomorrow.com/article/2016/07/robot-vision-vs-computer-vision-whats-the-difference/8484/, 21.2.2019.

[2] Huang, T.: *Computer Vision: Evolution and Promise*, 1996.

[3] Cognex: *Introduction to Machine Vision*, 2016.

[4] UKIVA: *Machine Vision Handbook*, 2007.

[5] N. N.: *Quick History of Machine Vision*, https://www.epicsysinc.com/blog/machine-vision-history, 21.2.2019.

[6] http://www.enciklopedija.hr/natuknica.aspx?ID=55546, 21.2.2019.

[7] https://www.servocity.com/hs-425bb, 21.2.2019.

[8] https://www.cis.hr/www.edicija/SSHprotokol.html, 21.2.2019.

[9] https://www.programiz.com/article/difference-compiler-interpreter, 21.2.2019.

[10] https://opencv.org/about.html, 21.2.2019.

PRILOG

I. Programski kod

II. CD-R disc